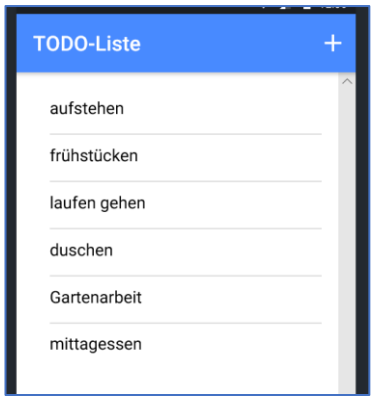
Übung Ionic – ToDo-Liste erstellen

Ziel:

- Mit „add“-Button neue Elemente aufnehmen
- Mit einem Wischen nach links ein Icon mit einem Papierkorb anzeigen lassen („sliding-item“)
- Dabei Ionic-Komponenten verwenden
- Löschen nach Klick auf den Papierkorb



1)Erstelle einen neuen Ordner „todo“ und lege ein neues Projekt (blank) an

- Z.B. am Desktop

- CMD öffnen und zum Ordner navigieren mit „cd“
  nun soll gleich hier der neue Name eingeben werden und die Vorlage „blank“ gewählt werden

```
C:\Users\heidi>cd desktop

C:\Users\heidi\Desktop>cd todo

C:\Users\heidi\Desktop\todo>ionic start todo blank
```

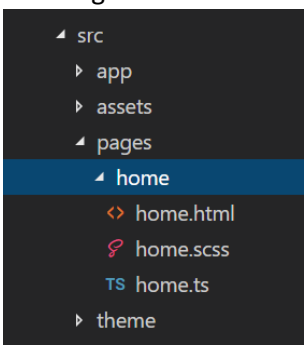
  Dann einmal „y“ und einmal „n“
  - install cordove….Yes
  - install pro SDK….No

- Mit dem Befehl „code .“ (code und Punkt) kann man gleich Visual Studio Code aufrufe und an befindet sich im gerade angelegten Orden.

```
C:\Users\heidi\Desktop\todo>code .
```

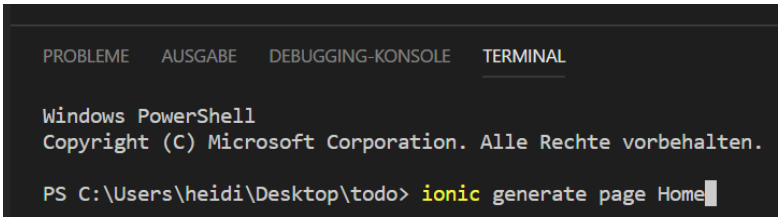
2)Vorbereitungen

- Lösche gleich wieder den Ordner „home“ und erstelle ihn neu

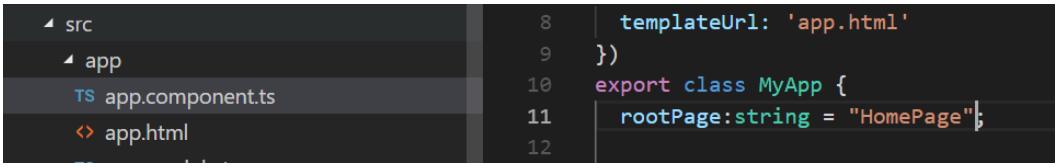


Ziehe das integrierte „Terminal“ von unten herauf um die „Home“ – Page neu zu erstellen mithilfe

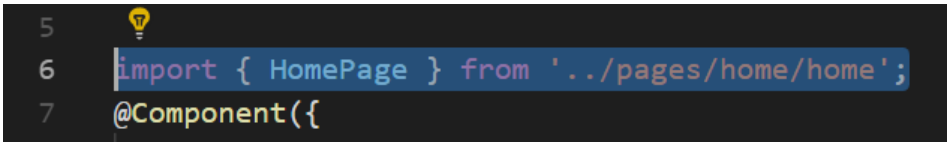
- ionic generate page Home



- Kleine Anpassungen:
  in src / app app.component.ts ändere Zeile 11 von „any“ auf „string“ und setzte die Anführungszeichen bei „HomePage“



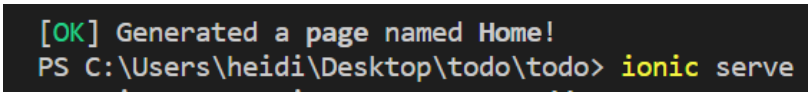
- In der gleichen Datei entferne Zeile 6 komplett, weil dieser Import nicht mehr nötig ist



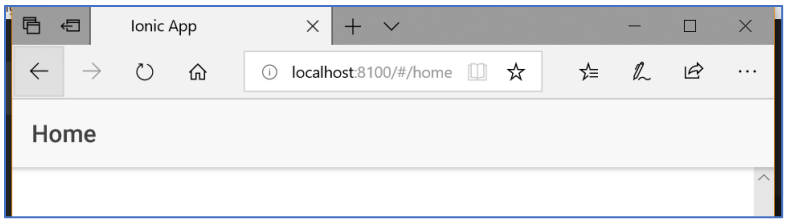
- Das gleiche in der Datei „app.moduls.ts“ in der Zeile 8
  Und darunter bei den „declarations“ die Zeile 13, das ja dann unterstrichen wird, wenn Zeile 8 vorher gelöscht wurde und bei den „entryComponents“ die Zeile 22.
- Beide Seiten speichern.

Zur Kontrolle nun einmal die App laden mit dem Code im Terminal

- ionic serve



Ergebnis:



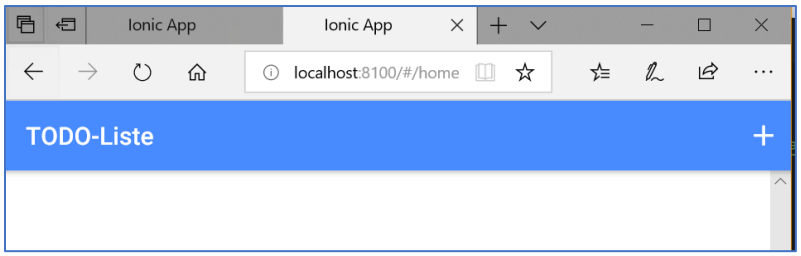
3)Button erstellen, um Eingabe aufzurufen

Erstellen einen Button rechts in der Navigation

Öffne „home.html“.
Ändere den Namen von Home auf „TODO Liste“ und gleich die Farbe in der Navbar auf „primary“. Darunter gib den Code für den Button ein. As Icon soll das „add“ dienen.

```
9   <ion-navbar color="primary">
10    <ion-title>TODO-Liste</ion-title>
11      <ion-buttons end>
12        <button ion-button icon-only>
13          <ion-icon name="add"></ion-icon>
14        </button>
15      </ion-buttons>
16  </ion-navbar>
```

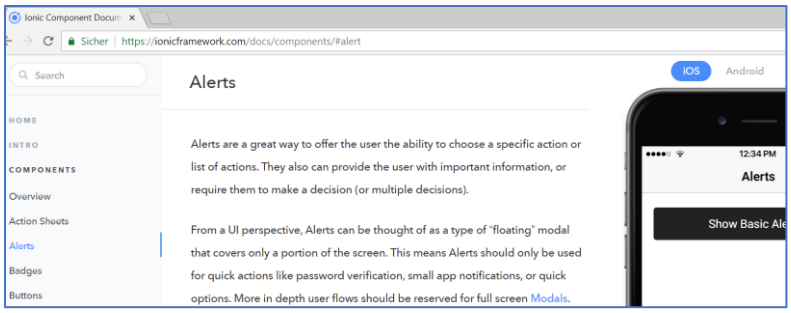
Ergebnis:



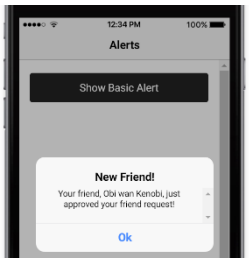
4)Ionic Komponenten verwenden für ein Alert-Fenster zur Eingabe

In der Website http://ionicframework.com unter „Developers“ und dann „UI Components“ findet man eine große Anzahl.

z.B. Alerts (für den Alert-Controller)



Dabei wird nach einem Klick auf einen Button ein Alert-Fenster geöffnet, um Daten aufzunehmen.



4a)Klick-Event

Dazu muss beim Button-Element ein Klick-Event eingebaut werden. Diese erhält eine neue Funktion, mit dem frei gewählten Namen „addItem“.

- (click)=“addItem()“

```
12    <button ion-button icon-only (click)="addItem()">
```

4b)Funktion hinterlegen

Danach muss diese Funktion hinterlegt werden, nämlich in der „home.ts“.

❖ Füge zuerst im „constructor“ diesen neuen Controller ein – die Schreibweise übernimm aus den beiden davorstehenden

```
16  .ass HomePage {
17
18  ictor(public navCtrl: NavController, public navParams: NavParams, private alertCtrl: AlertController)
19
```

❖ Darunter kommt nun die neue Funktion:

```
21    addItem() {
22
23    }
```

❖ Darinnen muss man nun der AlertController hinzugefügt werden: zum Erstellen und zum Präsentieren

```
21    addItem() {
22      this.alertCtrl.create().present();
23    }
```

❖ Was genau „create“ tun soll, folgt nun innerhalb von den Klammern von „create“

Zuerst muss man überprüfen, welche Elemente man überhaupt ausgeben kann. Daher klicke mit gedrückter STRG-Taste auf das Wort „create“.

```
224         */
225         create(opts?: AlertOptions): Alert;
226     }
```

Dann wird eine Zeile gezeigt, die mit „create“ beginnt. Hier klicke für Details wieder mit gedrückter STRG-Taste auf „AlertOptions“. Nun sieht man in hellblau, welche Elemente übergeben werden können.

Es gibt auch ein Element „input“ (Zeile 7). Dort gibt es das Element „AlertInputOptions[]“, welche deren Elemente gleich darunter zu finden sind.
Beispiel: name, placeholder und value

home.html ● | home.ts ● | *alert-options.d.ts* ×

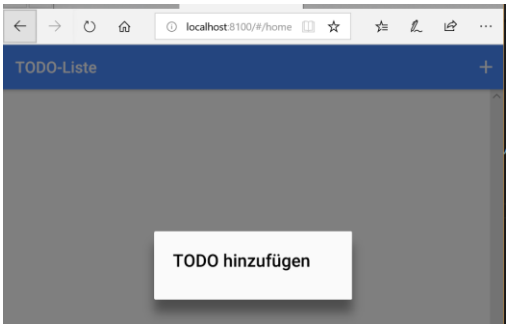
```
1   export interface AlertOptions {
2       title?: string;
3       subTitle?: string;
4       message?: string;
5       cssClass?: string;
6       mode?: string;
7       inputs?: AlertInputOptions[];
8       buttons?: (AlertButton | string)[];
9       enableBackdropDismiss?: boolean;
10  }
11  export interface AlertInputOptions {
12      type?: string;
13      name?: string | number;
14      placeholder?: string;
15      value?: string;
16      label?: string;
17      checked?: boolean;
18      disabled?: boolean;
19      id?: string;
20      handler?: Function;
21      min?: string | number;
22      max?: string | number;
```

❖ Zuerst muss man in die runden Klammern von „create()“ ein Paar geschwungenen Klammern setzten und den Code einfügen:

➢ Daher wird zuerst ein „title“ verwendet. Beachte die einfachen Anführungszeichen.

```
21  addItem() {
22    this.alertCtrl.create({
23      title: 'TODO hinzufügen'
24    }).present();
25  }
```

Ergebnis: das Alert-Fenster wird angezeigt



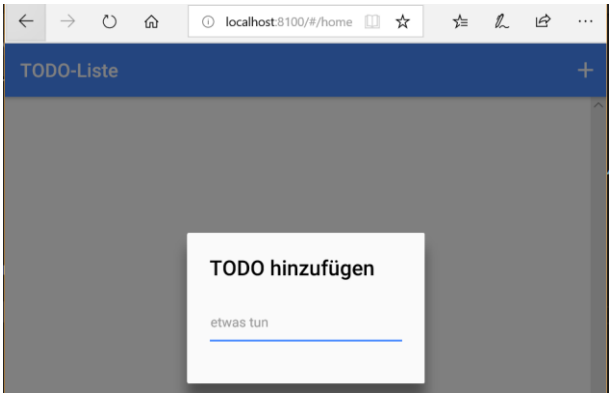
- Dann folgt ein „inputs“, welches aber mit eckigen Klammern verwendet wird, weil dies in den Optionen auch sichtbar ist, siehe Zeile 7:

```
7       inputs?: AlertInputOptions[];
```

Davor muss aber ein Beistrich nach dem „title“ gemacht werden.
Beachte die eckigen und geschwungenen Klammern und die einfachen Anführungszeichen

```
22    this.alertCtrl.create({
23      title: "TODO hinzufügen",
24      inputs: [
25        {
26          name: 'todo',
27          placeholder: 'etwas tun'
28        }
29      ]
30    }).present();
```

Ergebnis:



- Button unterhalb einfügen
  Beistrich nach dem Ende der eckigen Klammer von den „inputs“, weil ja noch ein Element folgt, nämlich „buttons“. Dies auch wieder mit eckigen und geschweiften Klammern:

```
24        inputs: [
25          {
26            name: 'todo',
27            placeholder: 'etwas tun'
28          }
29        ],
30        buttons: [{
31          |
32        }]
33      }).present();
34    }
```

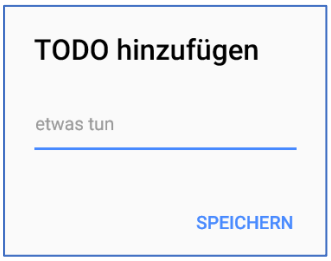
Nachsehen, welche Optionen möglich sind, die uns interessieren könnten, nämlich „text“ und „handler“ – mit dem was passieren kann, wenn was zurückkommen soll.

```
24  export interface AlertButton {
25      text?: string;
26      role?: string;
27      cssClass?: string;
28      handler?: (value: any) => boolean | void;
29  }
```

Verwende nun den „text“.

```
29          ],
30          buttons: [{
31            text: 'speichern'
32          }]
33        }).present();
34    }
```

Das Alert-Fenster sieht nun so aus:



5)Eingegebene TODO´s ausgeben

5a)Array anlegen

Dafür muss man ein Array anlegen, um die Eingaben speichern zu können. Dies oberhalb vom „constructor“.

Ein Namen ist frei zu wählen. Nimm „todos“. Der Type des Arrays ist „string“ und zuerst einmal leer. Es sind ja noch keine TODO`s vorhanden.

```
16  export class HomePage {
17
18    todos: string[] = [];
19
20    constructor(public navCtrl:
```

5b)Event Handler anlegen

Um die Eingaben zu erhalten, verwende unten im Button einen „handler“ mit „data“. Die „data“ werden in die TODO´s gepackt. Beistrich davor nicht vergessen.
In Zeile 35 wird das „todo“ genommen von Zeile 28, dem name: ‚todo‘

```
32          buttons: [{
33            text: 'speichern',
34            handler: data => {
35              this.todos.push(data['todo']);
36            }
37          }]
```

5c)Anzeige erstellen

Nun folgt noch die Anzeige der TODO´s.

Dies erfolgt in der „home.html“. Dort im „Content“.
Beginne mit einer Liste und einem Eintrag (item):

```
21  <ion-content padding>
22
23    <ion-list>
24      <ion-item>
25
26      </ion-item>
27    </ion-list>
```

Um alle Einträge anzeigen zu lassen, muss man im <item> eine Schleife laufen lassen.

- Dazu wähle: *ngFor =“let todo of todos“
  d.h. es werden alle „todo“ angezeigt, die in „todos“ stehen. Weil ja aus Zeile 35 alle „todo“ in das Array „todos“ in Zeile 18 gespeichert werden.

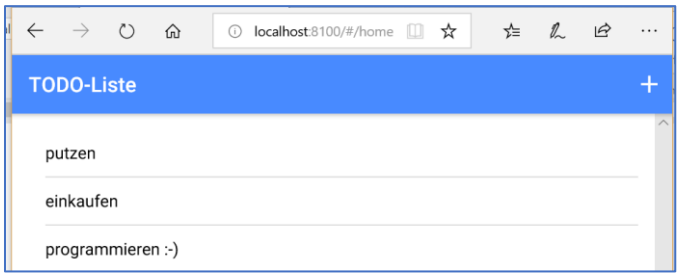
```
23    <ion-list>
24      <ion-item *ngFor="let todo of todos" >
25
26      </ion-item>
27    </ion-list>
```

Die echte Ausgabe ist zwischen den <item> nämlich in zwei geschwungenen Klammerpaaren:

- {{todo}}

```
23    <ion-list>
24      <ion-item *ngFor="let todo of todos" >
25        {{todo}}
26      </ion-item>
27    </ion-list>
```

Ergebnis:



6)Anzeigen wieder löschbar machen

Durch Klick auf einen Eintrag soll dieser gelöscht werden

Dafür erstelle einen „deleteItem“ in „home.ts“ unterhalb des „addItem“.

```
41  deleteItem() {
42
43  }
```

Das Item, das gelöscht werden soll wird in der runden Klammer übergeben:

```
41  deleteItem(todo: string) {
```

Das Klick-Event muss aber noch in „home.html“ erstellt werden, nämlich bei der Anzeige, wo auch die For-Schleife war. Denn dort wird man ja auf den Eintrag klicken. Der Name ist der,den man gerade erstellt hat (deleteItem). Übergeben wird dabei die entsprechende „todo“, die aus dieser Schleife kommt (gleiche Zeile nur vorher).

```
23  <ion-list>
24    <ion-item *ngFor="let todo of todos" (click)="deleteItem(todo)" >
```

6a)Info:

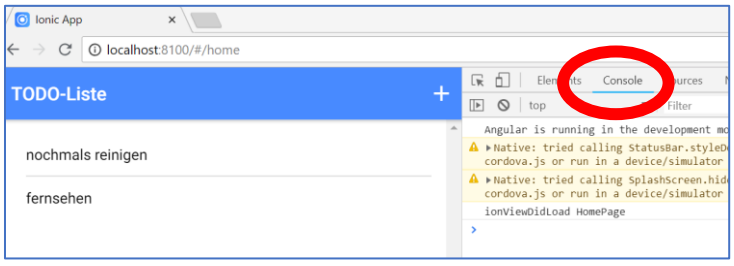
Für Testzwecke kann man sich das Item kurz in der Console ausgeben lassen

In der „home.ts“:

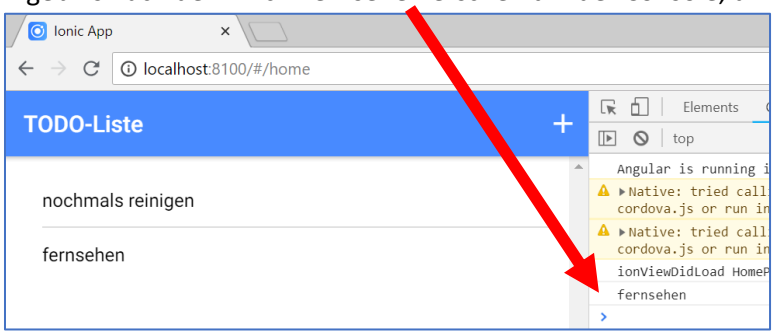
```
41  deleteItem(todo: string) {
42    console.log(todo);
43  }
```

Zu sehen in der Entwicklungsumgebung, die man in Chrome mit F12 aufruft. Dann im Reiter „Console“.

Zum Überprüfen gib nochmals neu zwei Werte ein und klicke danach auf einen zum Löschen.



Ergebnis nach dem Klick: fernsehen erscheint in der Console, d.h. es ist ok



Gelöscht wird noch nichts, da die Funktion noch nicht da ist.

6b)wirklich löschen

Entferne das „console.log) und schreibe

- var index = this.todos.indexOf(todo, 0);

```
41    deleteItem(todo: string) {
42      var index = this.todos.indexOf(todo, 0);
43    }
```

Hier wird der Index aus dem Todo-Array angezeigt.

Darunter muss nun der Index überprüft werden, ob er da ist, d.h. ob er größer als minus 1 ist. Minus eins deshalb, da ein Item immer mit 0 zu zählen beginnt. Wenn das wahr ist, dann wird das angeklickte Element auch wirklich gelöscht.

- if (index > -1) { dann wird gelöscht}

```
41    deleteItem(todo: string) {
42      var index = this.todos.indexOf(todo, 0);
43        if (index > -1){
44
45        }
46    }
```

- this.todos.splice(index, 1)

Das heißt, splice nimmt den index dafür, an welcher Stelle er löschen soll. Nämlich an der Stelle des „todo“ – siehe Zeile 42. Hier startet er somit. Dann wird hinterlegt, wie viele Einträge gelöscht werden sollen, hier genau einer.

```
41    deleteItem(todo: string) {
42      var index = this.todos.indexOf(todo, 0);
43        if (index > -1){
44          this.todos.splice(index, 1);
45        }
46    }
```

Fertig.

Nun testen.

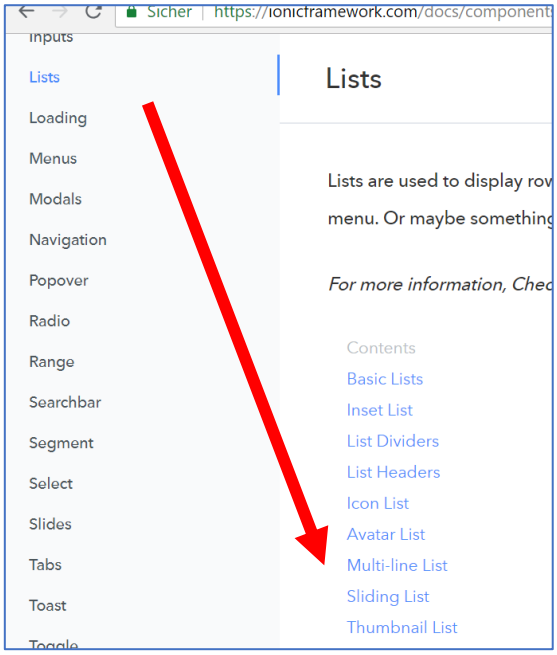
7)Sliding-List installieren

7a)Info auf der Website

https://ionicframework.com/docs/components/#overview

Developers / UI Components

und dann die „Lists“ – Sliding List



Inhalt:

Sliding items can be swiped to the left or right to reveal a hidden set of buttons.

Daher:

In der <list> der „home.html“ benötigt man nun eine Änderung. Die <ion-list> braucht <ion-itm-sliding>…

For more information, Check out the API docs.

```
<ion-list>
  <ion-item-sliding>
    <ion-item>
      <ion-avatar item-start>
        <img src="img/slimer.png">
```

```
23    <ion-list>
24      <ion-item-sliding>
25        <ion-item *ngFor="let todo of todos" (click)="deleteItem(todo)" >
26          {{todo}}
27        </ion-item>
28    </ion-item-sliding>
29    </ion-list>
```

Der Text in der {{todo}} kann eine Form erhalten, z.B. <h2>

```
  <h2>{{todo}}</h2>
</ion-item>
```

7b)Optionen eingeben:

Wenn man nach links (oder rechts) wischt, sollen Optionen erscheinen. Die Optionen stehen dann entweder links oder rechts. Bei uns sollen sie dann rechts stehen, nachdem nach links gewischt wurde. Daher benötigen wir die Version: **side="right"**

Es kann beides möglich sein, das nach links Wischen und das nach rechts wischen. Danach soll die Möglichkeit zu „löschen" gezeigt werden. **Wir wischen hier nach links.**

Kopiere aus der Docs-website oder schreibe folgenden Code für das Wischen nach links:

Hier neu: Zeile 28 - 33

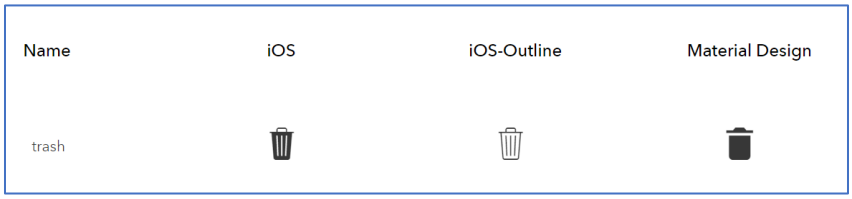
```
26          <h2>{{todo}}</h2>
27        </ion-item>
28        <ion-item-options side="right">
29          <button ion-button color="primary">
30            <ion-icon name="text"></ion-icon>
31            text
32          </button>
33        </ion-item-options>
34      </ion-item-sliding>
```

Danach muss der Klick von oben Zeile 25 hierher verlegt werden, da ja erst nach dem Wischen die Möglichkeit des Löschens erscheinen soll.

```
25      <ion-item *ngFor="let todo of todos" >
26        <h2>{{todo}}</h2>
27      </ion-item>
28      <ion-item-options side="right">
29        <button ion-button color="primary" (click)="deleteItem(todo)">
```

Das Icon soll auf einen „Mistkübel" (=trash) verändert werden und aber nur das Icon angezeigt werden, ohne Text. Daher „text" entfernen (Zeile 31) und in der Zeile, wo der Button ist, ein „icon-only" schreiben. Der Name für den Mistkübel lautet „trash"

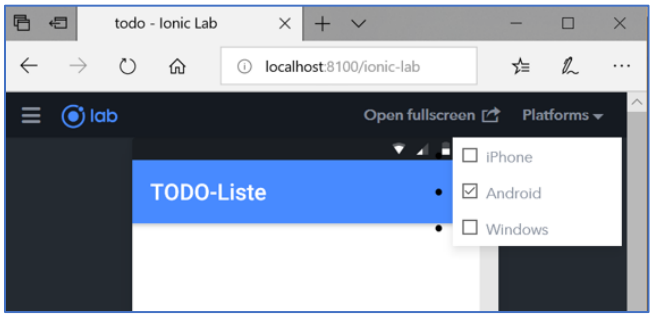
Zu finden unter „docs" – Ionicons:



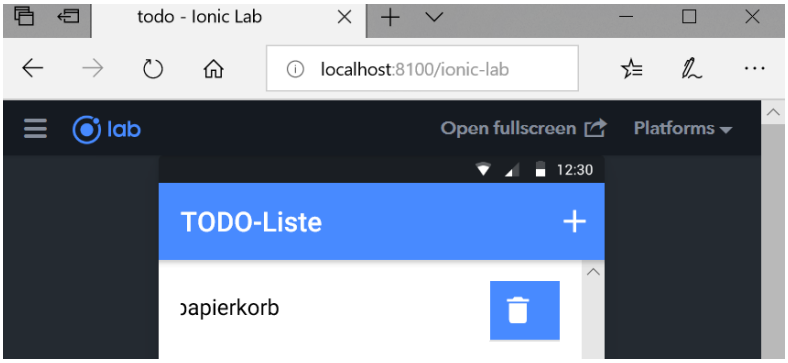
Ergebnis

```
<ion-item-options side="right">
    <button ion-button icon-only color="primary" (click)="deleteItem(todo)">
      <ion-icon name="trash"></ion-icon>
    </button>
```

Zum Testen muss man aber nun die Handy-Ansicht aufrufen. Das funktioniert im Terminal mit „ionic lab“. Dann stelle auf „Android“:



Test Beispiel: gib ein „test Papierkorb“ und wische nach links:



Das Löschen funktioniert aber erst, wenn sich die Schleife auf alles auswirkt und das ist erst der Fall, wenn man diese von <ion-item> nach <ion-item-sliding> verschiebt:

```
22
23  <ion-list>
24    <ion-item-sliding *ngFor="let todo of todos">
25      <ion-item  >
26        <h2>{{todo}}</h2>
```

Funktioniert!